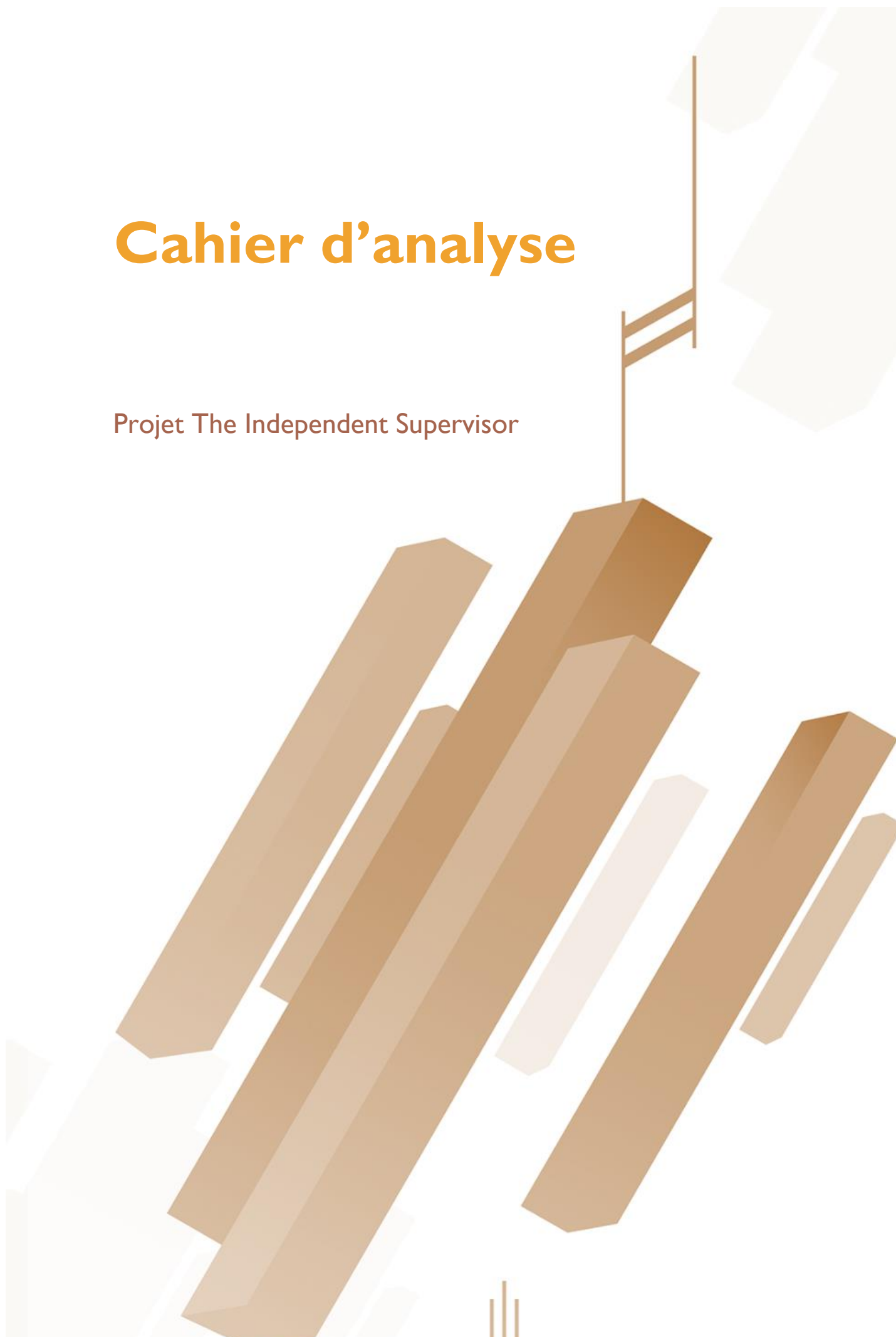


# Cahier d'analyse

Projet The Independent Supervisor



## Table des matières

I.	Introduction .....	4
1.	Rappel sur le projet .....	4
a)	Contexte du projet .....	4
b)	Objectif général du projet .....	4
2.	Rappel sur l'architecture .....	5
a)	Existant .....	5
b)	Objectif .....	6
II.	Analyse et Modélisation .....	7
1.	Modélisation UML .....	7
a)	Diagramme de bloc .....	18
b)	Diagramme de séquence .....	7
	Ajout d'une idée/suggestion .....	7
	Consultation d'une idée/suggestion .....	8
	Ajout d'une application .....	9
	Modification d'une configuration d'une application .....	10
	Suppression d'une application .....	11
	Démarrer une application .....	12
	Arrêter une application .....	13
	Mise à jour d'une application .....	14
	Supervision d'une application .....	15
	Redémarrage automatique en cas de crash d'une application .....	16
	Arrêt automatique en cas de manque de ressources .....	17

Commentaire	Date de modification
Création du document	13/11/2018
Prévision de la partie Analyse & Modélisation	26/11/2018
Ajout de plusieurs diagrammes de séquence	27/11/2018
Ajout des derniers diagrammes de séquence	03/12/2018
Descriptions des technologies qui correspondent aux besoins	08/01/2019

// WIP

Présentation des technologies

Choix des technologies

Diagramme de classes

Mock up interface Web

Description de l'hébergement (@IP -> site web)

Arborescence des fichiers

## I. Introduction

### 1. Rappel sur le projet

#### a) Contexte du projet

Dans le cadre de notre formation en école d'ingénieurs, nous devons réaliser un projet de fin d'études durant notre 5<sup>e</sup> année. Ce projet débute mi-Septembre 2018 correspondant à notre début de période école pour cette dernière année, pour finir à la fin de notre période école, pendant le mois de Février 2019. Étant donné que le projet se déroule en même temps que la période école, il n'est pas possible de travailler à plein temps sur le projet, une demi-journée sera donc consacrée au minimum chaque semaine.

The Independent Gamers est une équipe de joueurs amateurs, cofondé en 2013 par Brandon SIMON-VERMOT, cette équipe partage l'idée de fournir une ambiance divertissante en proposant diverses activités et la possibilité de rencontrer des joueurs réguliers sur une panoplie de jeux divers. The Independent Supervisor est le fruit d'une idée venant du chef de l'équipe de joueurs amateur qui possède un serveur dédié, sur lequel plusieurs serveurs de jeux sont installés.

- Brandon SIMON-VERMOT : Maitre d'œuvre
- Aneur SOUKHAL : Encadrant de projet & maitre d'ouvrage

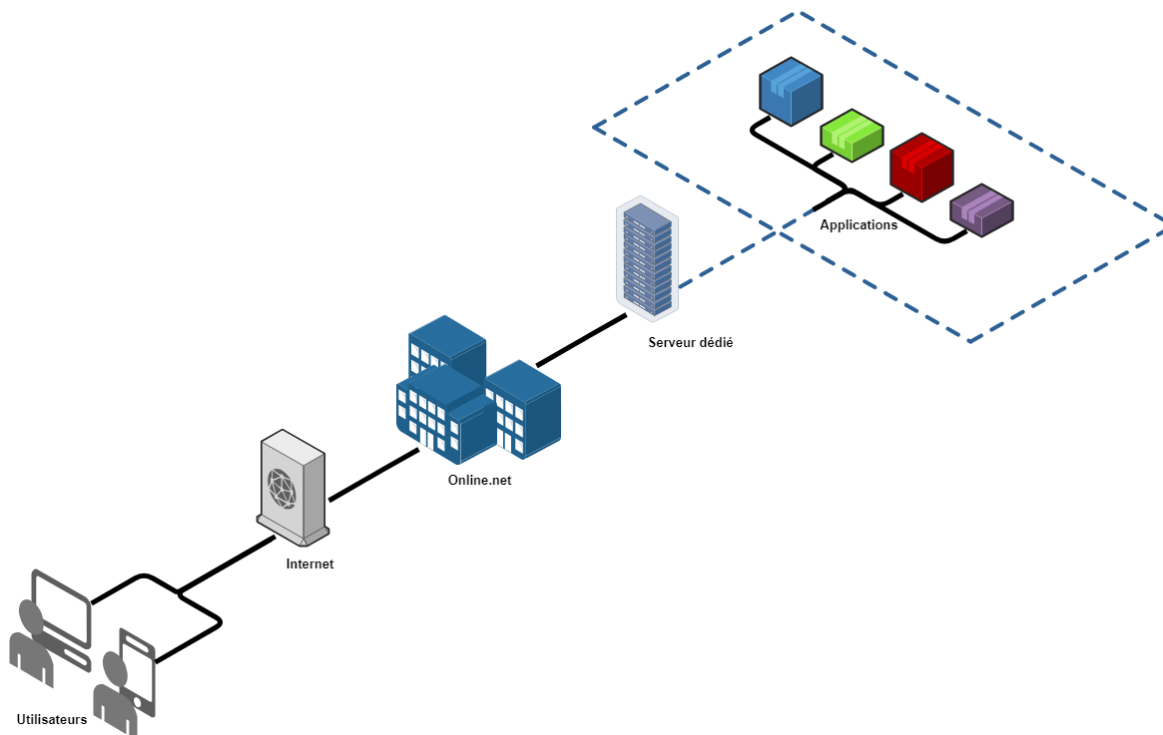
Pour plus d'informations de manière globale, la lecture du cahier de spécification peut s'avérer utile à la bonne compréhension du projet.

#### b) Objectif général du projet

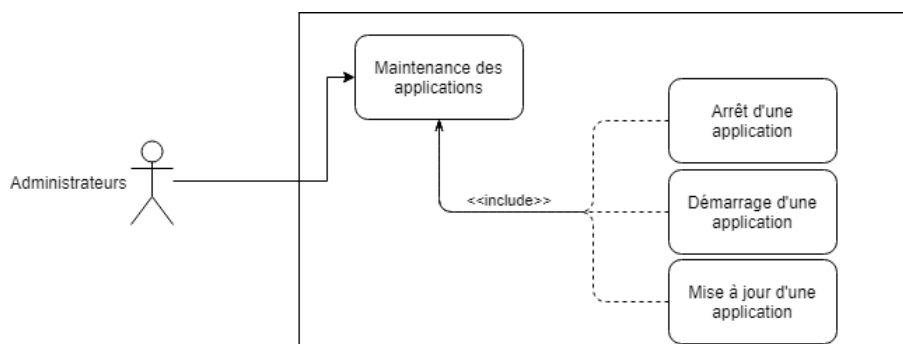
Le besoin de ce projet est de superviser les serveurs de jeux et offrir la possibilité de les mettre à jour automatiquement, de les maintenir. Pour ce faire, des lancements, des arrêts d'application (via des commandes enregistrées associées à une application par exemple) seront exécutés automatiquement pour remplacer la maintenance manuelle, ce qui est fait actuellement. Ce projet part de zéro, aucun site Web et les outils liés (serveur Web, BDD, ...) ne sont déjà installés sur le serveur dédié, seuls les serveurs de jeux sont actuellement présents. Cette application web permettra donc de pouvoir exécuter une application ou l'arrêter grâce aux informations enregistrées par les administrateurs de l'application Web. Notamment les informations autour de chaque serveur de jeu (emplacement du dossier du serveur, commande de lancement, commande de sauvegarde), qui ici dans le cadre de ce projet seront les applications du parc soumis à la supervision.

## 2. Rappel sur l'architecture

### a) Existant

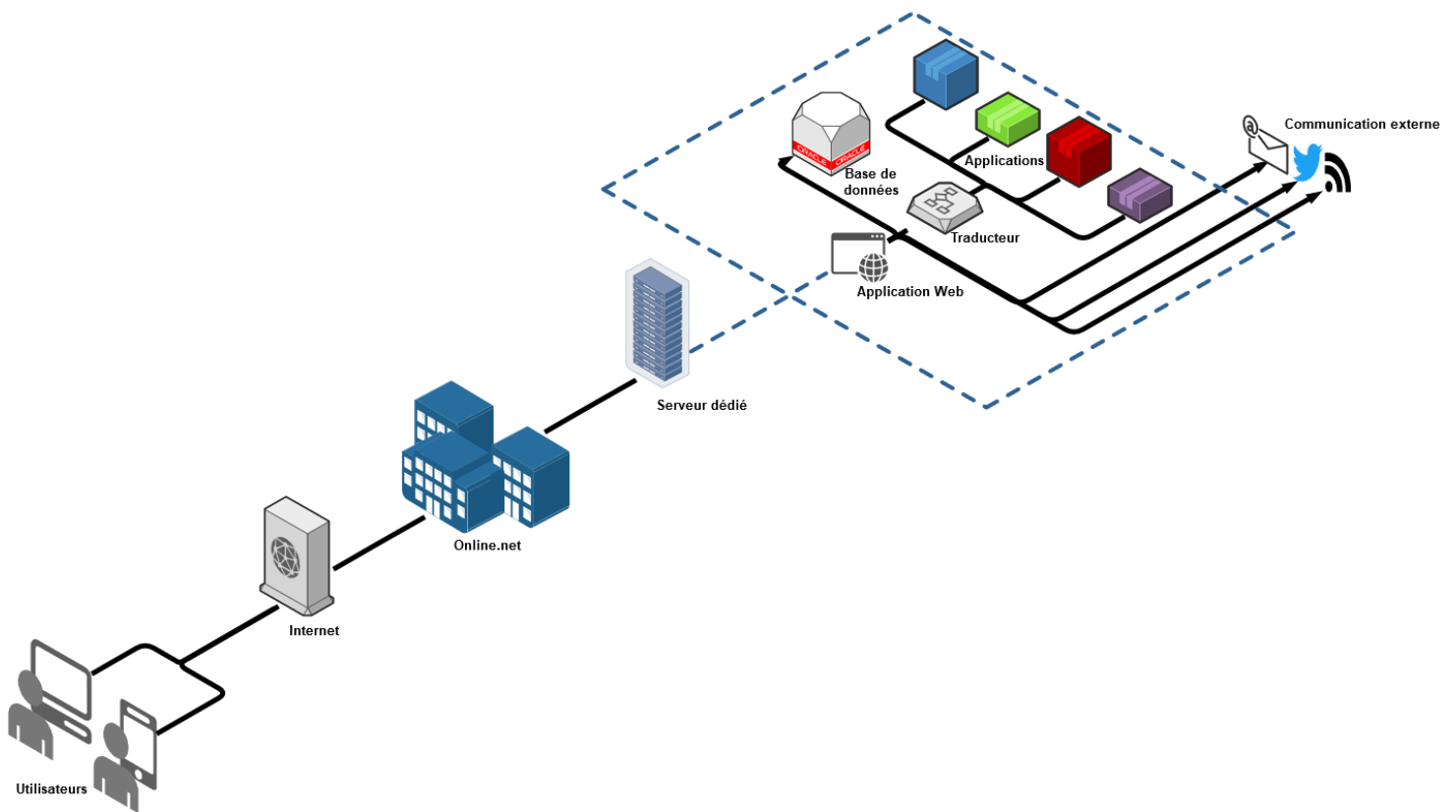


Chaque serveur de jeu, chaque application, présent sur le serveur dédié, qui est actuellement sous Ubuntu, a donc besoin d'être maintenu, d'être redémarré s'il y a une erreur, via des lignes de commandes. Cela demande du temps et un accès au serveur via une application de télémaintenance ou via un hyperviseur, actuellement KVM.



Le cas d'utilisation actuel, ne prends pas en compte les joueurs, puisque leurs seules interactions se limitent à la connexion et la communication avec les applications qui sont démarrées sur le serveur dédié. Dans le cadre actuel, il n'y a qu'un seul administrateur qui puisse se connecter sur le serveur dédié pour réaliser la maintenance du parc d'application.

## b) Objectif



Pour mieux représenter le projet, ci-dessus se trouve un schéma comprenant les différents acteurs et éléments constituant l'architecture générale du système envisageable à la fin du projet. Dans un premier temps nos acteurs seront nos utilisateurs (administrateurs ou non), ils devront avoir une connexion Internet pour accéder à l'application Web, cette application est hébergée sur un serveur dédié appartenant au fournisseur Online.net.

L'application Web étant le cœur du projet, elle est associée à différents éléments, une base de données contenant les différentes configurations de nos applications, dans le cadre de ce projet ce seront des serveurs de jeux. Ces applications sont différentes ce qui indique qu'elles peuvent avoir un moyen de communication différent, d'où l'intérêt d'avoir un élément qui nous servira de traducteur, dans le but de transmettre correctement l'information souhaitée.

Cette même application Web permettra de communiquer des informations sur l'état actuel des serveurs de jeux, pour cela il devra être possible de communiquer via différents moyens afin d'en informer nos utilisateurs, via par exemple le réseau social Twitter ou encore un flux RSS.

L'application Web devra pouvoir envoyer des mails aux administrateurs pour les prévenir en cas de problème mais aussi pour toute demandes et suggestions envoyés par nos utilisateurs via l'application Web.

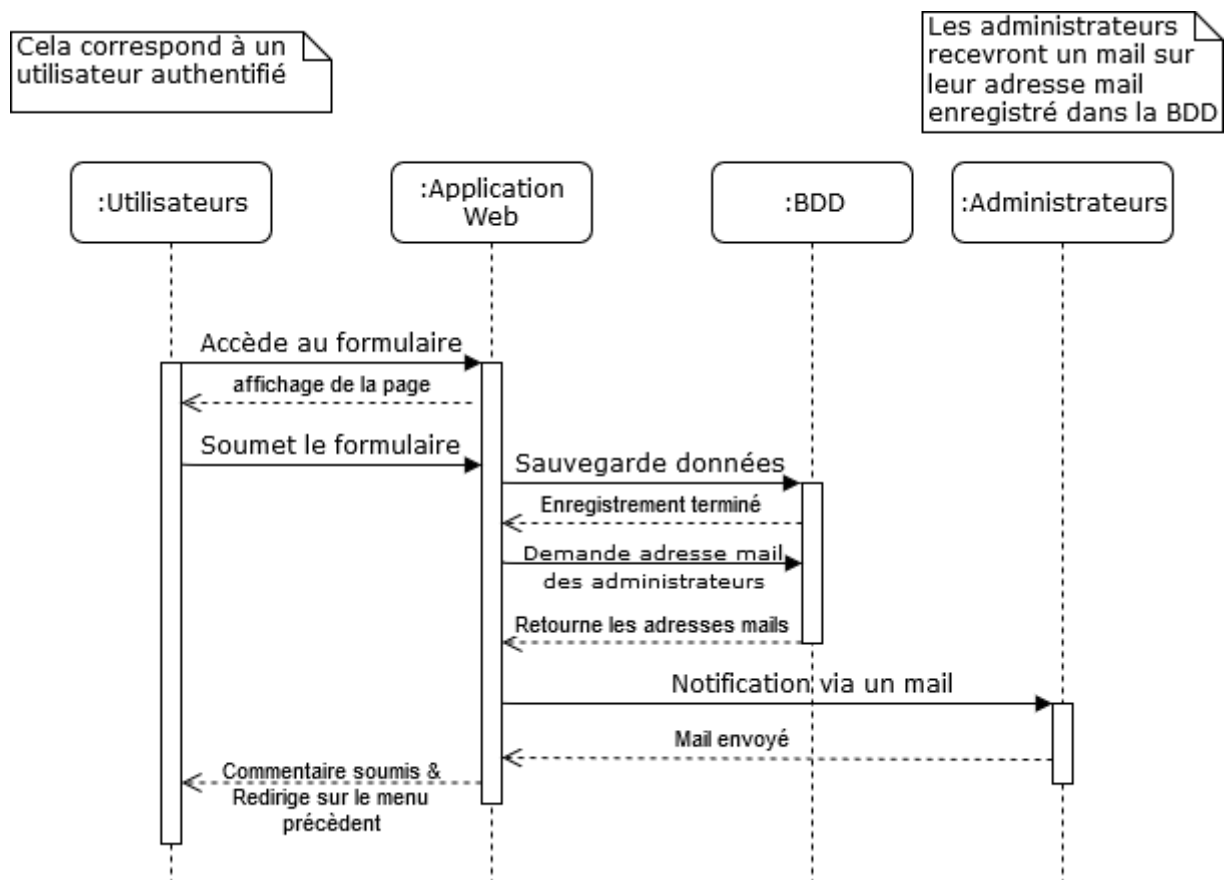
## II. Analyse et Modélisation

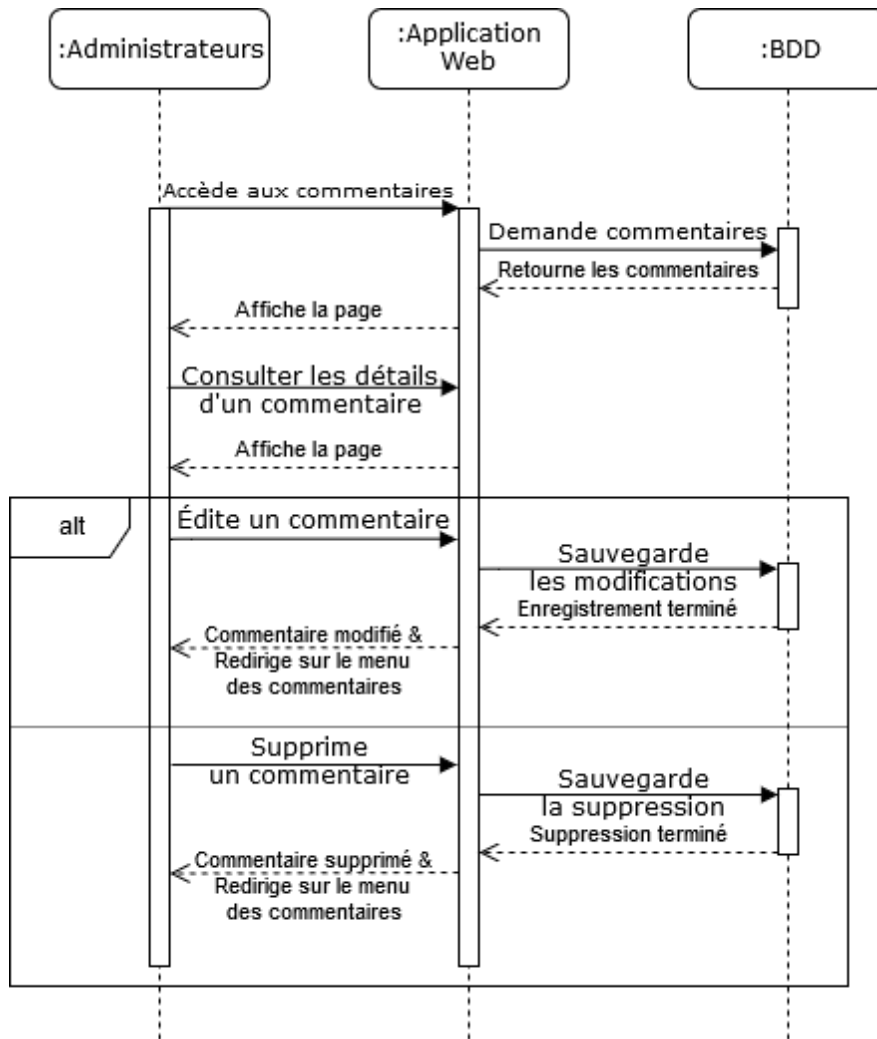
### 1. Modélisation UML

#### a) Diagrammes de séquence

Pour tous les diagrammes présents ci-dessous, une authentification est nécessaire au préalable. Cette authentification est simple, on demande un nom d'utilisateur/adresse mail en tant qu'identifiant pour un utilisateur et un mot de passe associé.

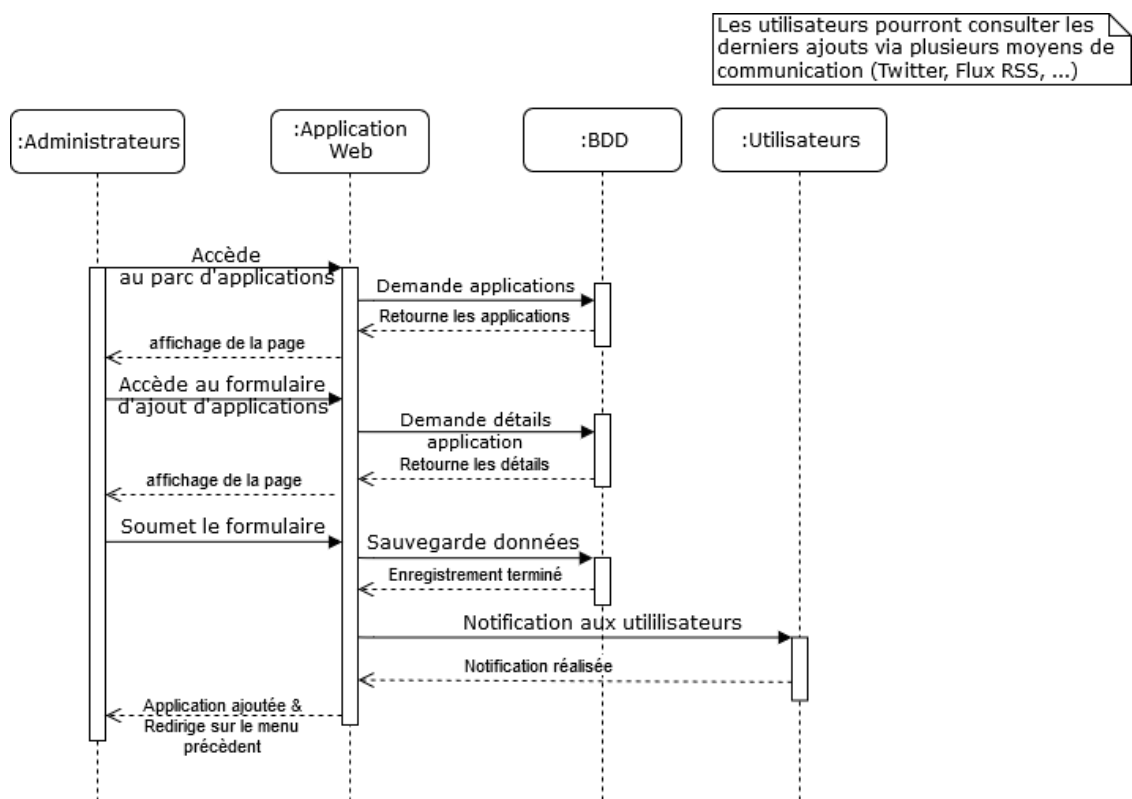
*Ajout d'une idée/suggestion*

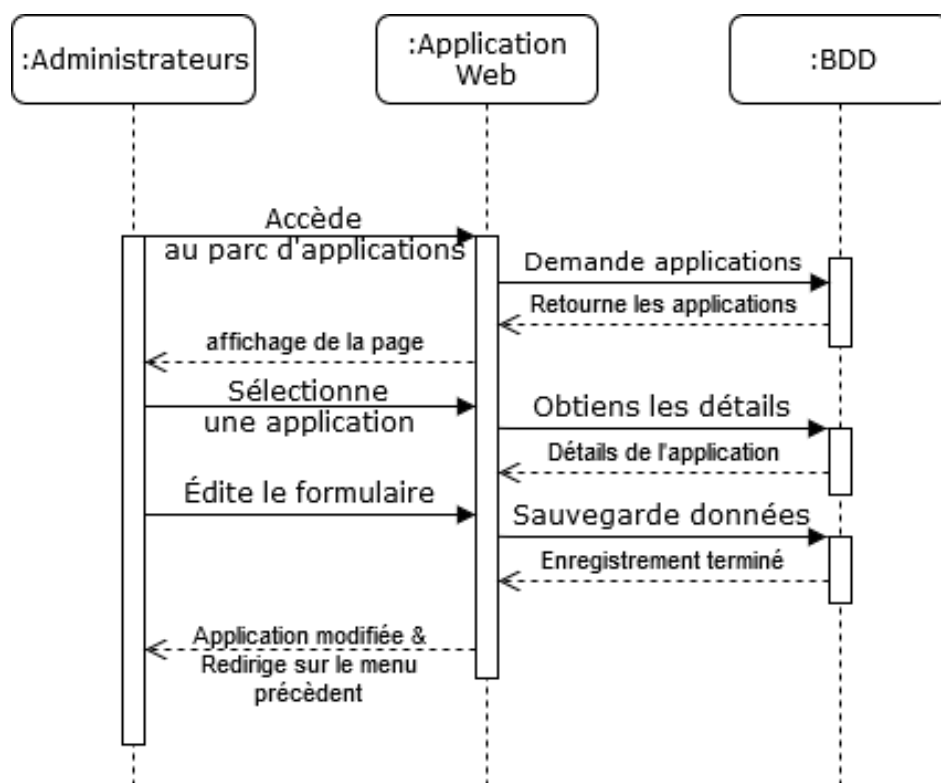


*Consultation d'une idée/suggestion*

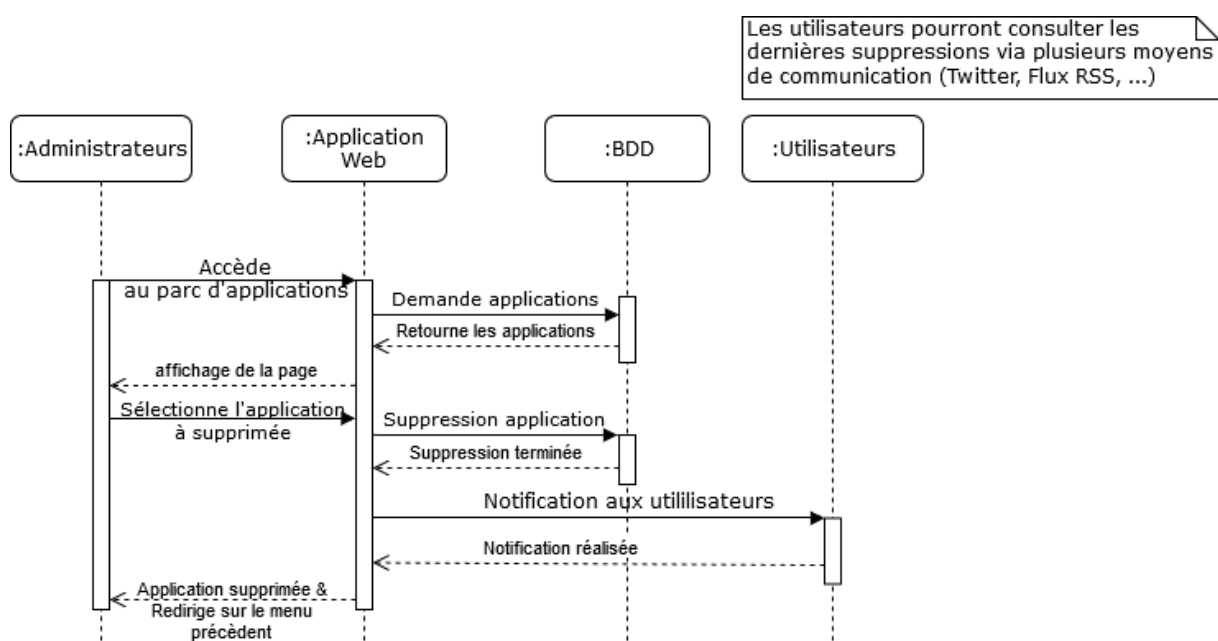


## Ajout d'une application

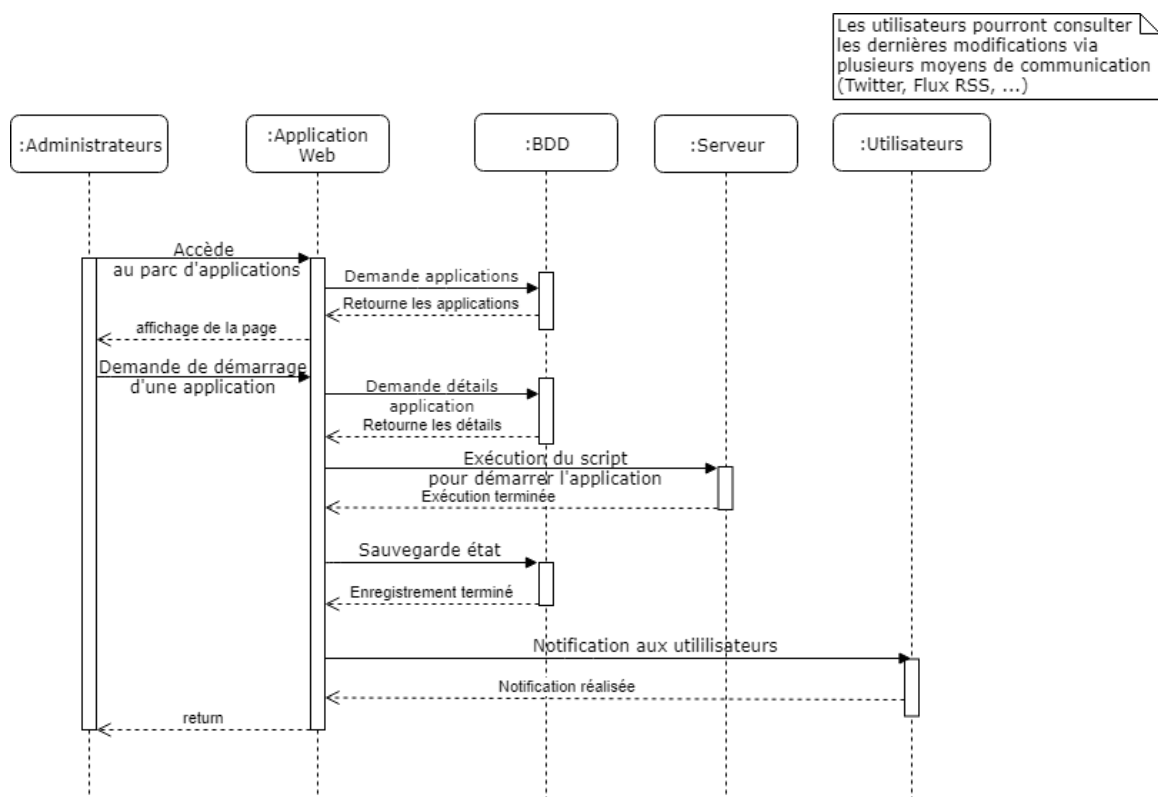


*Modification d'une configuration d'une application*

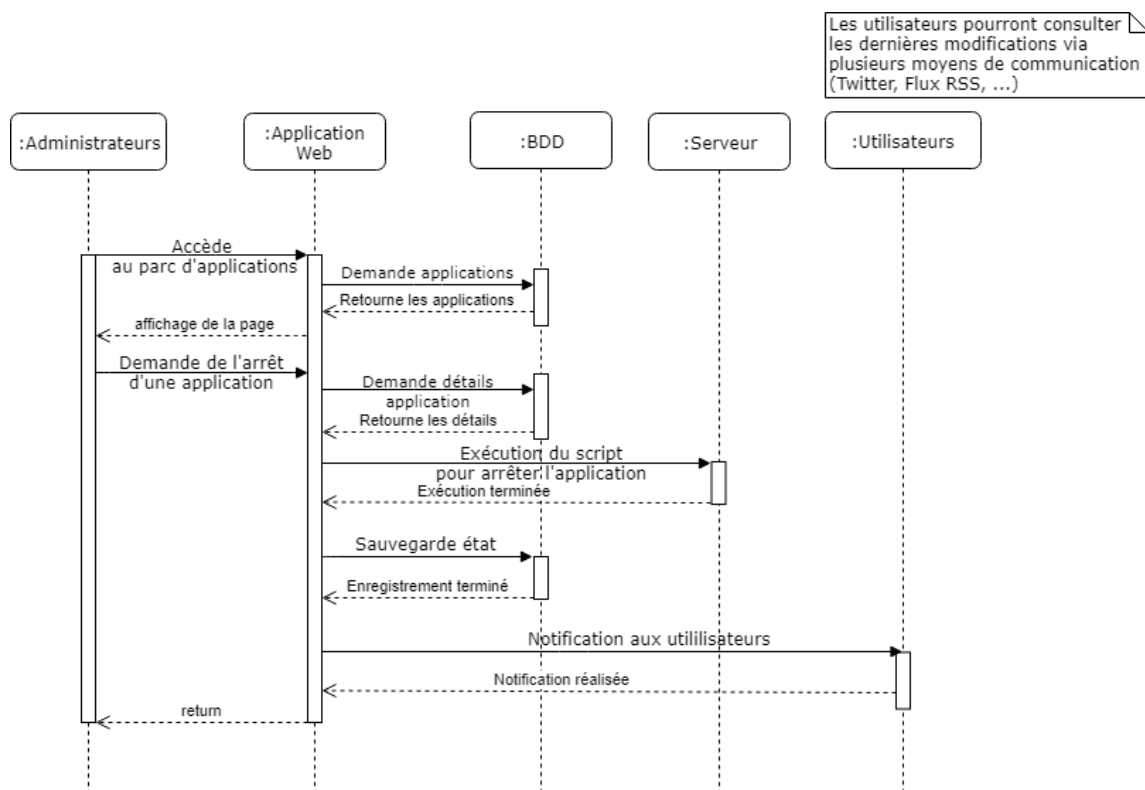
## Suppression d'une application



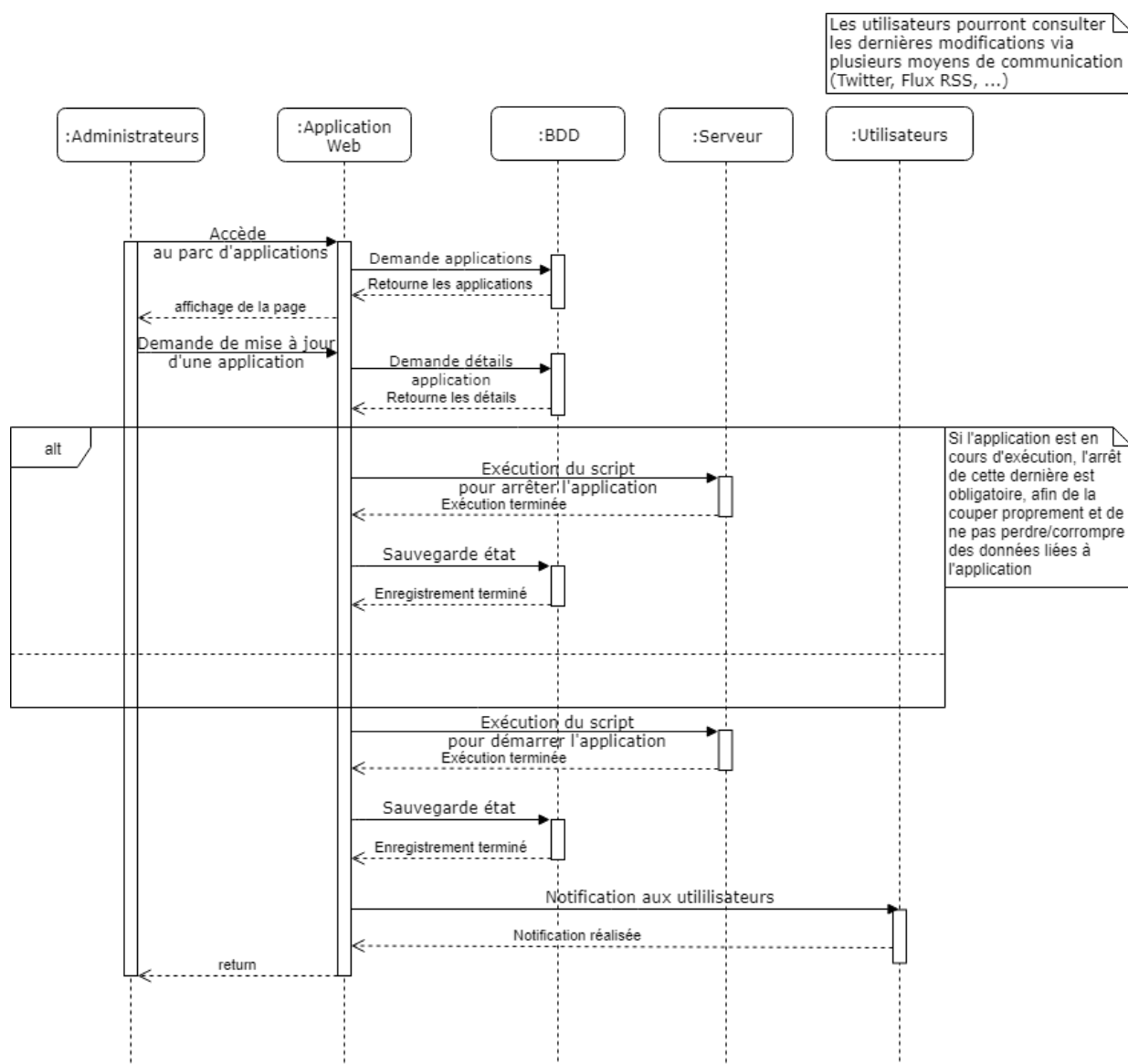
## Démarrer une application

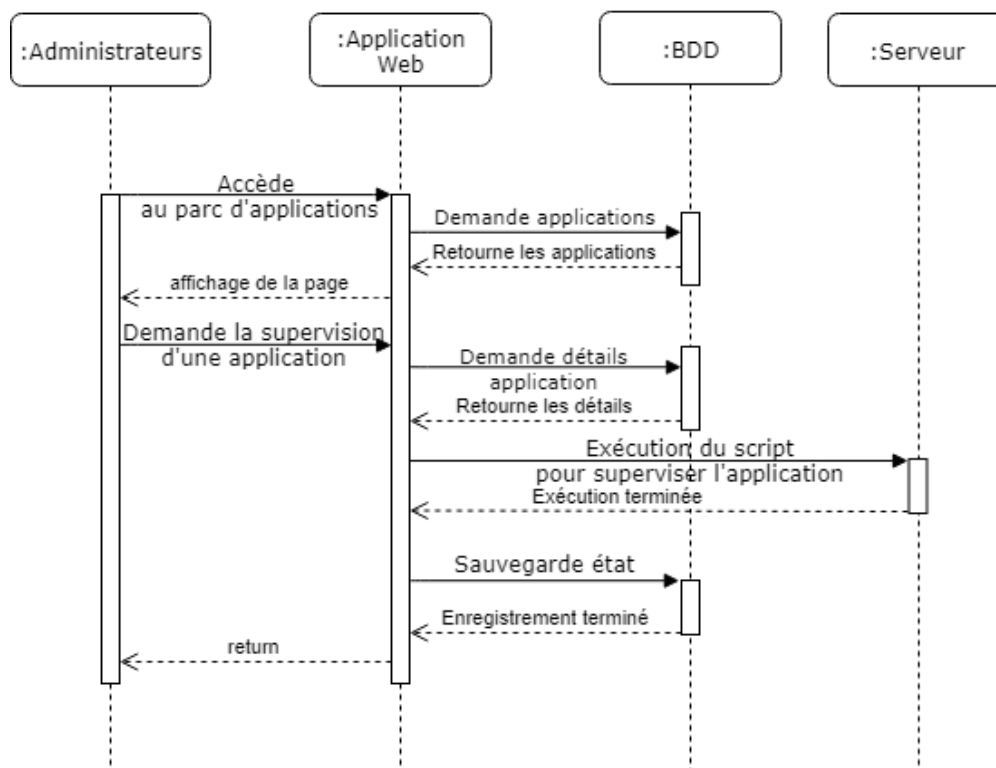


## Arrêter une application

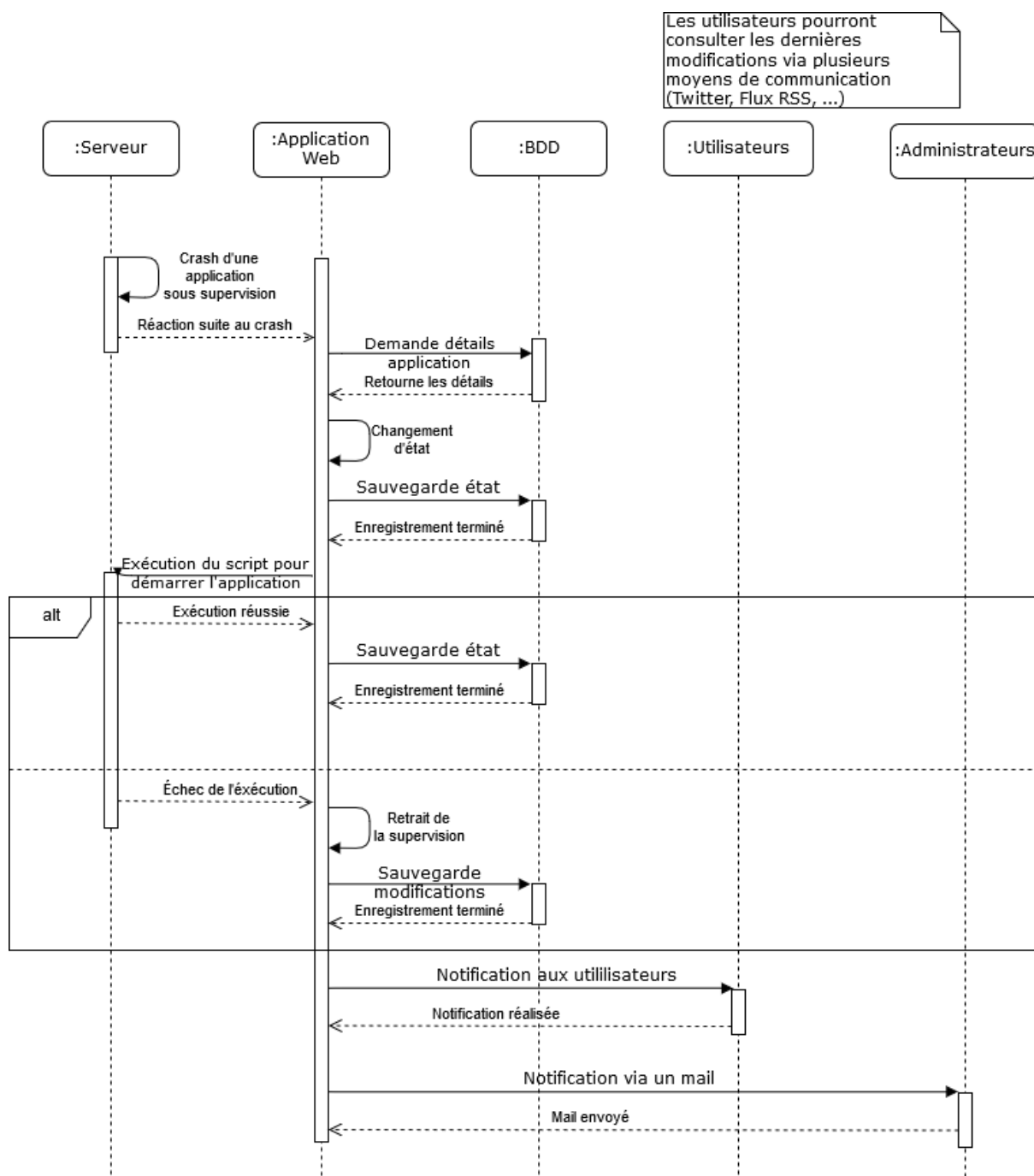


## Mise à jour d'une application



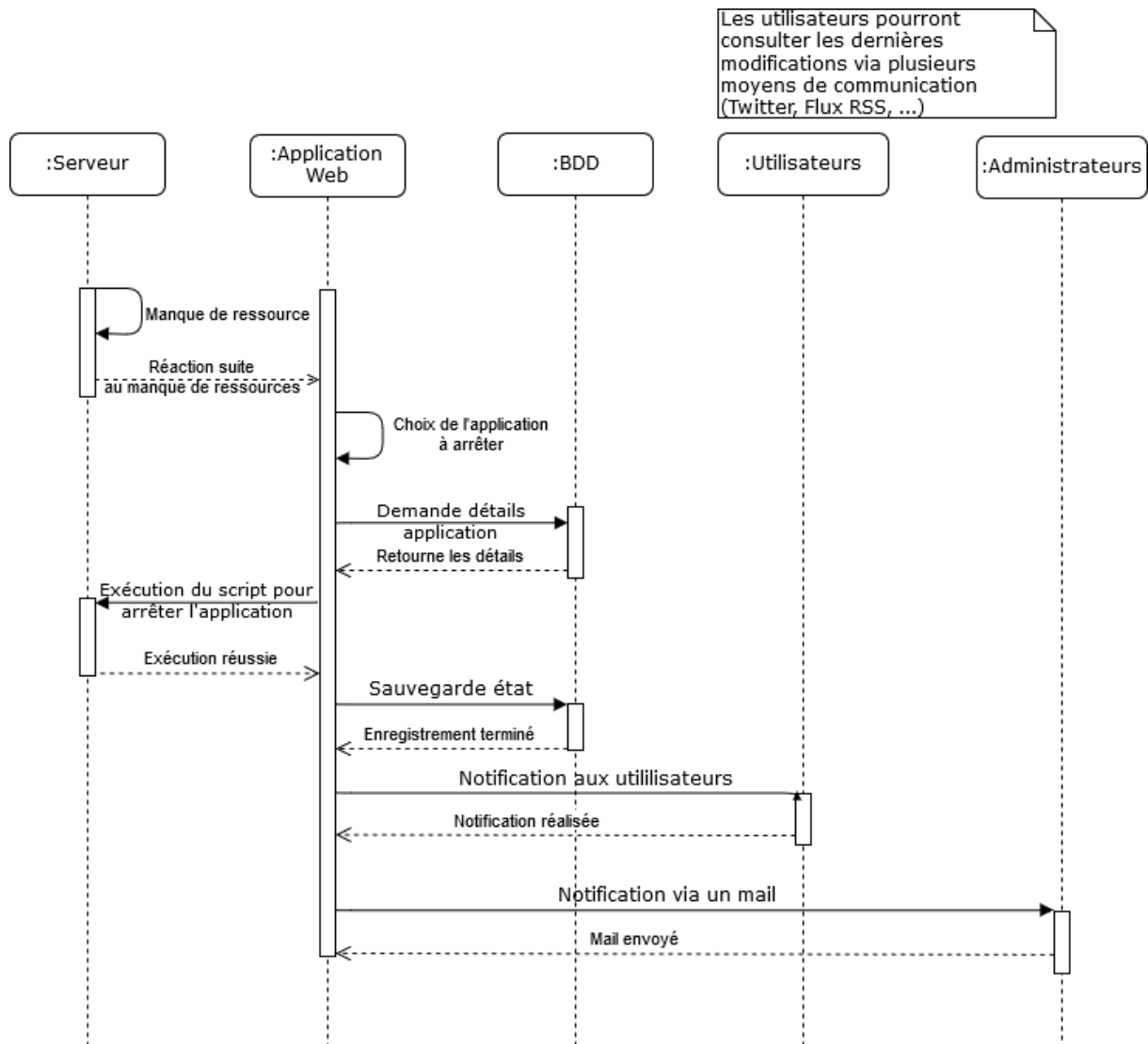
*Supervision d'une application*

## Redémarrage automatique en cas de crash d'une application





## Arrêt automatique en cas de manque de ressources



b) Diagramme de classes

## 2. MEAN Stack



## a) MongoDB



### *Base de données*

Une base de données est une collection d'informations organisées afin d'être facilement consultables, gérables et mises à jour facilement. Au sein d'une base de données, les données sont organisées en lignes, colonnes et tableaux. Elles sont indexées afin de pouvoir facilement trouver les informations recherchées à l'aide d'un logiciel informatique. À chaque modification de la base de données de nouvelles informations sont ajoutées ou sont mises à jour, voir éventuellement supprimées.

Elles peuvent se charger de créer, de mettre à jour ou de supprimer les données souhaitées. Elles peuvent permettre d'effectuer des recherches parmi les données qu'elles contiennent sur demande de l'utilisateur.

Les bases de données sont stockées sous forme de fichiers ou d'ensemble de fichiers sur tout type d'appareil de stockage. Les bases de données dites traditionnelles sont organisées par champs, enregistrements et fichiers. Un champ est une seule pièce d'information (ex : ville => TOURS). Un enregistrement est un ensemble de champs (ex : prénom => Brandon, nom => SIMON-VERMOT, ville => TOURS, ...). Ces enregistrements sont stockés sous formes de fichiers.

Comme parfait exemple, un répertoire téléphonique peut représenter un fichier qui compose la base de données. Ce répertoire est constitué d'un ensemble d'enregistrements, dont chaque enregistrement regroupe trois champs : nom, adresse et numéro de téléphone.

## Orienté documents

MongoDB est un système de base de données orienté documents. À l'inverse avec un système de base de données « traditionnel », dites base de données relationnelle, les données sont stockées par ligne dans des tables. Et il est souvent nécessaire de faire des jointures sur plusieurs tables afin de tirer des informations assez pertinentes de la base.

Dans un système orienté documents, les données sont modélisées sous forme de document sous un format JSON.

On ne parle plus de tables et d'enregistrements mais de collections et de documents. Ce système de gestion de données nous évite ainsi de faire des jointures de tables car toutes les informations propres à un certain donnée sont stockées dans un même document.

Les documents sont les unités de base dans une base MongoDB. Ils sont équivalents aux objets JSON et sont comparables aux enregistrements d'une table dans une base de données relationnelle.

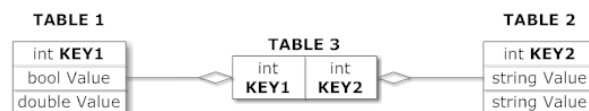
Tout document appartient à une collection et a un champ appelé `_id` qui identifie le document dans la base de données. Le champ `_id` est donc unique afin d'identifier correctement un document.

MongoDB enregistre les documents sur l'espace de stockage sous un format BSON (JSON binaire).

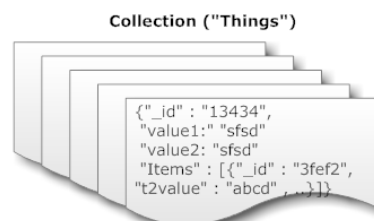
Une collection est un ensemble de documents, l'équivalent d'une table en relationnel. Contrairement aux bases de données relationnelles, les champs des documents d'une collection sont libres et peuvent être différents d'un document à un autre. Le seul champ commun est obligatoire est le champ `"_id"`.

Néanmoins pour que la base soit maintenable, il est préférable d'avoir dans une collection des documents de même type.

### Relational Model



### Document Model



b) Angular

### c) Node.JS



Node.js est un environnement permettant d'exécuter du code JavaScript hors d'un navigateur.

Son architecture est modulaire et événementielle. Il est fortement orienté réseau en possédant pour les principaux systèmes d'exploitation (Unix/Linux, Windows, macOS) de nombreux modules réseau (dont voici les principaux par ordre alphabétique : DNS, HTTP, TCP, TLS/SSL, UDP). De ce fait, il remplace avantageusement, un serveur web tel qu'Apache.

Créé par Ryan Lienhart Dahl en 2009, cet environnement est devenu rapidement très populaire pour ses deux qualités principales :

- Sa légèreté (en corollaire de sa modularité).
- Son efficacité induite par son architecture monothread (en corollaire de la gestion événementielle que propose nativement l'environnement JavaScript).

Intégrer Node.js dans le développement d'applications web participe donc à la logique actuelle de rendre les opérations d'accès aux données les moins bloquantes possible (pour dépasser la problématique dite du « bound I/O » selon laquelle, avant toute autre cause, la latence globale d'une application est due au temps de latence des accès aux données).

Node.js permet donc, pour les applications web, de créer des serveurs extrêmement réactifs.

d) Express



### 3. Serveur dédié

#### a) Hébergement

b) Arborescence des fichiers

*Serveur Web (Front / Back End)*

*Parc d'applications*